DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Primer Trabajo Práctico

14 de Abril de 2010 Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Bianchi, Mariano	92/08	bianchi-mariano@hotmail.com
Brusco, Pablo	527/08	pablo.brusco@gmail.com
Di Pietro, Carlos Augusto Lyon	126/08	cdipietro@dc.uba.ar

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Ejercicio 1

1.1. Introducción

El primer problema de este trabajo práctico consistió en dar un algoritmo que sea capaz de decir, dada una secuencia de números, la cantidad de números mínima a eliminar de la misma para que cumpla con cierta propiedad. En este caso, la propiedad que debía cumplir la secuencia modificada era que la misma sea unimodal. Una secuencia es unimodal si la misma cumple que es creciente desde el primer elemento hasta cierta posición y desde dicha posición es decreciente hasta el final[1].

Además, se requirió que la complejidad del algoritmo que resolviera el problema antes mencionado, debía ser estrictamente menor que $O(n^3)$.

Se pensaron varias formas de encarar y resolver este problema, finalmente se aplicó la técnica de *programación dinámica*. En las secciones siguientes, daremos una explicación detallada de cómo se implementó el algoritmo, cuál es su complejidad temporal y cómo se comportó frente a distintos valores y tamaños de entrada.

1.2. Explicación

Desde el comienzo resultó bastante difícil encontrar dónde se podía aplicar el *principio de optimalidad* para resolver este problema utilizando *programación dinámica.* Luego de mucho observar y releer, se pudo intuir que este problema, podía ser visto como una combinación de otro problema similar: encontrar la subsecuencia creciente/decreciente más larga. Básicamente, el razonamiento fue el siguiente:
Si para cada posición i de la secuencia sabemos:

- Longitud de la subsecuencia creciente más larga desde el principio hasta i (que contenga al elemento i-ésimo)
- Longitud de la subsecuencia decreciente más larga desde i hasta el final (que contenga al elemento i-ésimo)

entonces podemos decir cuál es el largo de una secuencia unimodal que tiene como "pivoteü objeto más grande al i-ésimo de la secuencia. Finalmente, sólo bastaría saber cuál es el "pivote"que maximice esa longitud, ya que conocer la longitud de la subsecuencia unimodal más larga es equivalente a encontrar la cantidad mínima de elementos a eliminar para transformar la secuencia dada en una secuencia unimodal, que es el problema que necesitabamos resolver.

Entonces, veamos como conseguimos la subsecuencia creciente más larga desde el principio hasta el pivote con el siguiente pseudocodigo:

[1] Debe ser estrictamente creciente/decreciente

void ascenso(v: vector)
Complejidad: *O(n^2)*
for $i = 0; i <= tam(v); i++$ **do**
| ascenso[i] ← maximoAsc(ascenso,v,i) + 1;
end

Donde ascenso es un arreglo donde se va guardando , en la posicion i, la longitud del mayor ascenso hasta i. La función maximoAsc(ascenso,v,i) indica, dentro del arreglo ascenso, la maxima longitud de un ascenso hasta la posicion i-1 del arreglo tal que cumpla la siguiente propiedad:

Si j es el indice del ascenso donde se encuentra el resultado, tiene que ocurrir que:

$$(\forall k \in [0...i), v[k] < v[i]) ascenso[j] > ascenso[k]$$

para lo cual utilizamos el siguiente algoritmo:

nat maximoAsc(ascenso: vector(nat), v: vector(int),i: nat)
Complejidad: *O(i)*
var res: nat res ← 0
for $k = 0; k < i; k++$ **do**
| **if** *v[j] < v[i] and res < ascenso[j]* **then** res ← ascenso[j]
end

1.3. Análisis de la complejidad del algoritmo

1.4. Detalles de Implementación

1.5. Resultados

1.6. Debate

1.7. Conclusiones

2. Ejercicio 2

2.1. Introducción

2.2. Explicación

Primero, algunas definiciones:

Def$_1$:
Un grafo G es fuertemente orientable si existe una asignación de direcciones a los ejes del conjunto de ejes del grafo G tal que el digrafo resultante es fuertemente conexo.

Def$_2$:
un puente, **arista de corte** o istmo es una arista que al eliminarse de un grafo incrementa el número de componentes conexos de éste. Equivalentemente, una arista es un puente si y sólo si no está contenido en ningún ciclo.

Teorema [Robbins, 1939] :
Un grafo conexo G es fuertemente orientable si y solo si G no tiene puentes (Demostración en el anexo).

Con esté teorema podemos ver que si encontramos al menos 1 puente en nuestro grafo, significa que no podremos orientarlo como queremos, y de lo contrario, si encontramos que no hay ningun puente, podremos orientarlo. Por lo tanto, el algoritmo utilizado realiza exactamente esa comprobación. Veamos como trabaja:

comprobación(Grafo G)

Complejidad: $O(n^3)$
var eje : int $\leftarrow$ 0
var n : int $\leftarrow$ cantNodos(G)
while $eje \leq m$ **do**
    k $\leftarrow$ RecorridoSinEje(eje,G)
    if $k \neq n$ **then return** no se puede
    eje $\leftarrow$ eje + 1
end
return fuertemente conexo

RecorridoSinEje(eje,G) es una funcion que recorre el grafo G (con BFS o DFS) sin utilizar la arista eje y retorna la cantidad de nodos visitados, k. Como la forma de recorrer utilizada, solo recorre nodos conectados a la raiz (es decir, al nodo donde comienza el recorrido), quiere decir que el resultado k va a ser n (la cantidad total de nodos de G) si G es conexo. De lo contrario, si k es menor que n, estamos en presencia de 2 componentes conexas (o más en el caso de $eje = 0$). Por lo tanto, el eje sacado, era un puente.

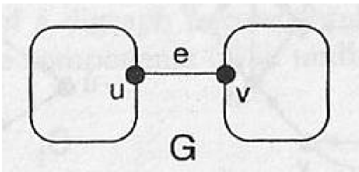
Aclaración:
Cuando eje = 0, representa, recorrer a G completo con todos sus nodos. En este punto podria pasar que G no sea conexo y esta función devolvería el resultado correspondiente.

2.3. Anexo: Demostraciones

Teorema [Robbins, 1939]:
Un grafo conexo G es fuertemente orientable si y solo si G no tiene puentes.

Demostración: $\rightarrow$)

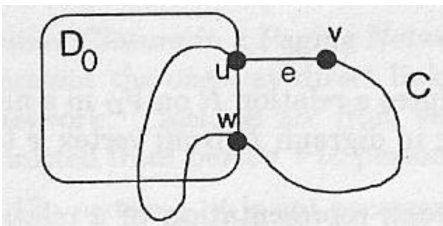
Utilizando el contrareciproco, supongamos que el grafo G tiene una arista de corte e que une a los vertices u y v. Entonces el unico camino entre u y v o v y u en el grafo G es e (ver figura). Por lo tanto para cualquier asignacion de direcciones, el nodo cola(e) nunca va a poder ser alcanzada por el nodo cabeza(e) (ver cuadro 1).



Cuadro 1:

Demostración: $\leftarrow$)

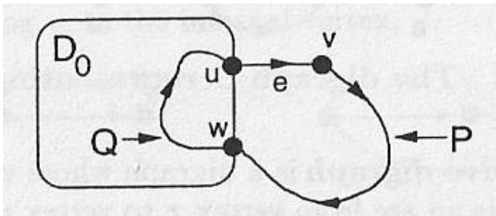
Supongamos que G es un nodo conexo sin aristas de corte. Por esto toda arista en G cae en un circuito de G. Para hacer que G sea fuertemente orientable, empezaremos con cualquier circuito (D_0) de G y dirigiremos sus ejes en una dirección (obteniendo un circuito dirigido). Si el ciclo D_0 contiene todos los nodos de G, entonces la orientación esta completa (ya que D_0 es fuertemente conexo). De lo contrario, hay que elegir cualquier eje e uniendo a un vertice u en D_0 y a un vertice v en $V_G - V_{D_0}$ (ese eje existe ya que G es conexo). Sea C = $\langle u, e, v, ..., u \rangle$ un ciclo que contiene al eje e, y sea w el primer vertice luego de v en C que cae en el ciclo D_0 (ver cuadro 2).



Cuadro 2:

A continuación, direccionamos los ejes de este camino entre v a w, obteniendo el camino dirigido $v-w$ que llamaremos P. Luego direccionamos el eje e desde u hasta v y consideramos el digrafo D_1 que se forma agregando el eje dirigido e a D_0 y todos los vertices y ejes dirigidos del camino P. Como D_0 es fuertemente conexo, entonces hay un camino dirigido de w a u (Q) en D_0 (ver cuadro 3). La concatenación de P con Q y el eje dirigido e forman un circuito simple direccionado que contiene u y los nuevos vertices de D_1. (si los vertices u y w son el mismo, entonces P satiface ser un circuito simple dirigido)

Por lo tanto, el vertice u y todos estos nuevos vetices son mutuamente alcanzables en D_1 y además u y cada vectice del digrafo D_0 tambien son mutuamente alcanzables, y, por lo tanto , el digrafo D_1 es fuertemente conexo. Este proceso puede continuar hasta que el digrafo D_1 para algun $l \geq 1$, contenga todos los vertices de G. En este punto, cualquier asignacion de direcciones hacia las aristas sin direccion restantes completaran la orientación de G, puesto que contendrá el digrafo fuertemente conexo D_l como subdigrafo.



Cuadro 3:

3. Ejercicio 3

3.1. Introducción